Efficient Approximate Parallel Prefix Adder Design

Jihwan Lim, Yuseok Lee, Donghun Lee and Hoyoung Yoo
Dept. of Electronics Engineering
Chungnam National University
Daejeon, Korea
jhlim.cas@gmail.com, yslee.cas@gmail.com, dhlee.cas@gmail.com, hyyoo@cnu.ac.kr

Abstract

Parallel Prefix Adder (PPA) was proposed as a structure to improve the performance of serial adders by parallelizing the carry operation. While it increases area and power consumption, PPA offers a faster addition operation compared to serial adders, making it a suitable solution for high-performance accelerator design. In domains that utilize PPA, ongoing research aims to reduce its area and power consumption. Approximate Computing is a methodology introduced in response to this demand, sacrificing some bit accuracy to reduce both area and power consumption. This paper introduces generally applicable approximate techniques to reduce the area and power consumption of PPA and quantitatively determines the most efficient techniques for Approximate PPA through experiments. We propose an Approximate PPA using an efficient approximate technique combination and demonstrate the design's superiority by comparing its performance with the previously proposed AxPPA and circuit area saving, power saving, performance based on ASIC design. Our proposed design demonstrates a 52.74% circuit area saving, 42.74% power saving, a 53.15% reduction in MAE , and a 53.6% reduction in MRED effect compared to AxPPA designs.

Keywords: Approximate computing(AxC), approximate technique, approximate adder(AxA), parallel prefix adders(PPA), approximate parallel prefix adder(AxPPA).

1. Introduction

Adders are widely used in the field of digital systems, serving a variety of applications, not just limited to basic arithmetic operations like multiplication and decimal addition but extending to more advanced applications in core and accelerator. An adder typically employs the Full-Adder as its basic design, which is composed of two Half-Adders. Design a Multi-bit Adder using multiple Full-Adders. Various structures for Multi-bit Adders have been proposed, such as Ripple Carry Adder(RCA), Carry Lookahead Adder(CLA), Carry Select Adder(CSLA), and Carry Skip Adder(CSKA). The simplest structure, RCA, connects Full-Adders in a linear fashion, making it the most area-efficient Multi-bit Adder. However, it suffers from significant delay due to the propagation of the carry signal. To resolve this problem, structures such as CLA, CSLA, and CSKA were proposed, each with its own method of carry calculation[1]. To more effectively resolve the delay problem, the Parallel Prefix Adder (PPA) structure was introduced. PPA consists of three stages, pre-processing, prefix-processing, and post-processing. Different PPA has been proposed based on the configuration of the prefix-processing stage [3-6]. PPA offers the advantage of minimal delay compared to traditional serial adders. However, it requires logic for parallel carry calculation, which can result in suboptimal performance in terms of circuit area and power efficiency compared to serial adders. To resolve this problem with each adder design, research has been conducted on applying Approximate Computing(AxC) to adder to optimize accuracy and reduce circuit area and energy consumption, resulting in Approximate Adder(AxA). The range of AxC application includes the transistor level, full-adder level, multi-bit adder level, and PPA level. We conduct research on Approximate PPA with AxC applied at the PPA level. A structure with AxC applied to PPA is proposed in [2], known as AxPPA. This paper analyzes the limitations of AxPPA and introduces the Efficient Approximate PPA(EAxPPA).

2. Background

A. Parallel Prefix Adder

PPA typically consists of three stages, pre-processing, prefix-processing, and post-processing, with each stage employing different circuit configurations to perform specific functions. The first stage, pre-processing, is responsible for encoding the input operands a and b. It includes logic for generating the carry g and logic for propagating the carry p. The boolean equations for the p and g of the i-th bit are given by (1), (2).

$$p_i = a_i \oplus b_i, \tag{1}$$

$$g_i = a_i \cdot b_i. \tag{2}$$

The second stage, prefix-processing, groups p and g to generate the final carry signal over several steps. prefix-processing is composed of an operation block

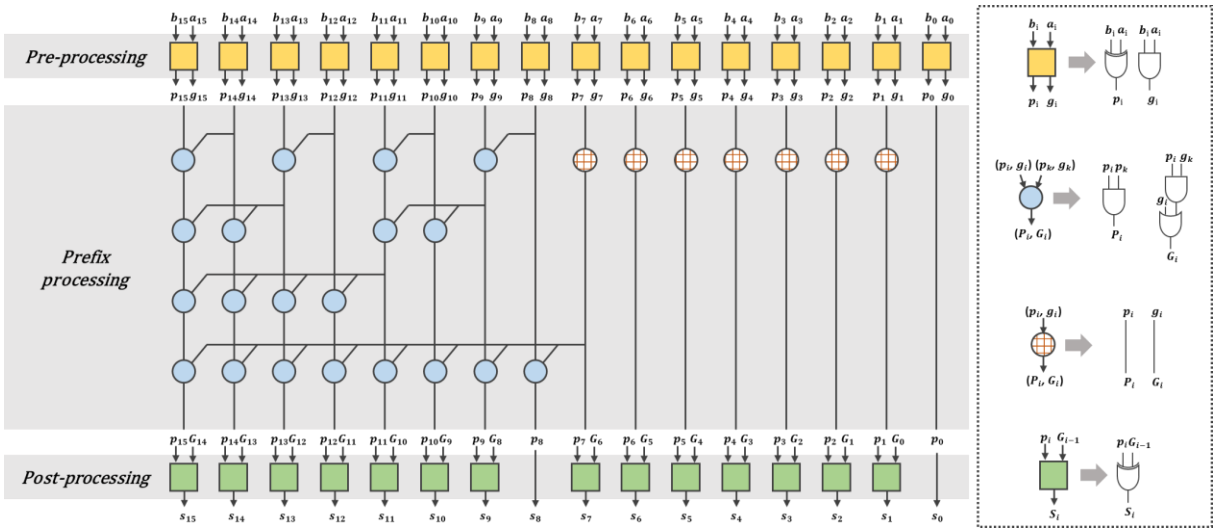

Figure 1. AxPPA structure

called the Prefix Operator (PO). The boolean equations for the PO are provided by (3), (4).

$$G_i = g_k \cdot (p_i + g_i), \tag{3}$$

$$P_i = p_i \cdot p_k. \tag{4}$$

G is generated based on the previous node's g_k and expressions for p_i and g_i for the i-th bit. P is generated based on p_i for the i-th bit and the previously calculated p_k. The outputs P and G from the PO are then connected as inputs to other POs. Unlike traditional serial adders, POs in PPA are connected in parallel over multiple stages to compute the carry quickly. The last stage, post-processing, combines the p from the pre-processing stage and the individual bit carries G calculated in the prefix-processing stage to generate the final sum S for operands. The boolean equation for the i-th bit is given by (5).

$$S_i = p_i \oplus G_{i-1}. \tag{5}$$

PPA performance varies based on the configuration of the PO, leading to the proposal of different PPA types such as Kogge-Stone(KS) PPA[3], Brent-Kung(BK) PPA[4], Sklansky(SK) PPA[5], and Ladner-Fischer(LF) PPA[6].

B. Approximate Technique

The approximate techniques used in the design of Approximate PPA can be broadly defined as three main techniques:

1) Elimination Technique: The elimination technique involves removing gates from the existing operator and connecting the input and output of the operator using wires.

2) Constant Technique: The constant technique entails removing gates from the existing operator and connecting constant values, typically cte-0(1'b0) and cte-1(1'b1), to the output.

3) Simplification Technique: The simplification technique involves replacing the gates in the existing operator with other area-efficient gates(and, nand, or, nor, xor, nxor).

C. AxPPA Design

Approximate PPA refers to a modified structure of the existing PPA, where approximate techniques are applied to improve the area and power efficiency issues inherent in traditional PPA. One prominent structure is the AxPPA proposed in [2], which introduces modifications to address these inefficiencies. Figure 1 shows the structure of AxPPA. In the case of AxPPA, the key proposal involves divide the entire PPA into two parts, and apply elimination technique to low part POs to enhance efficiency.

3. Proposed Method

A. Limits of AxPPA

AxPPA has several limitations that need to be considered. First, the approximate technique is only applied to POs. Ignoring the impact on performance when applied to other operators, apart from POs, is not advisable. Second, results are generated by passing through multiple operators from input to output. Therefore, basing the selection and application of an approximate technique on the error rate of a single PO is insufficient. Additionally, providing only a single technique lacks a comparative analysis of applying different techniques. In this paper, we consider to address these limitations and design an our proposed Approximate PPA.

B. Optimal Approximate Technique Combination

We find the optimal approximate technique combination applicable to PPA, and apply it to our design. At each stage, a different approximate technique is applied, and a comprehensive Z-Score

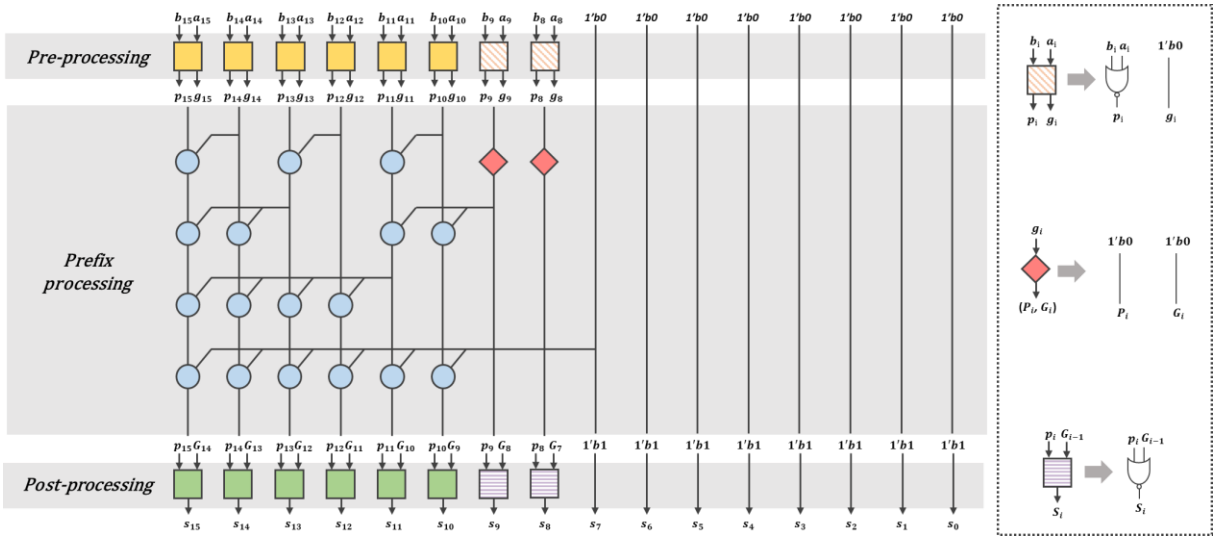

Figure 2. EAxPPA structure

Table 1. Circuit Area by design.

Design	Area
wire	0.468
cte-0,1	0.351
and	0.585
nand	0.468
or	0.585
nor	0.468
xor	1.053
nxor	0.936

Table 2. Top Z-Score by Combination.

Combination	Z-Score
[nor, cte-0, cte-0, cte-0, nor]	**-5.2674**
[nor, cte-0, cte-1, cte-0, nor]	-5.2646
[nor, cte-1, cte-0, cte-0, nor]	-5.2111
[nor, cte-1, cte-1, cte-0, nor]	-5.2074
[nand, cte-1, cte-0, cte-1, nand]	-5.1737

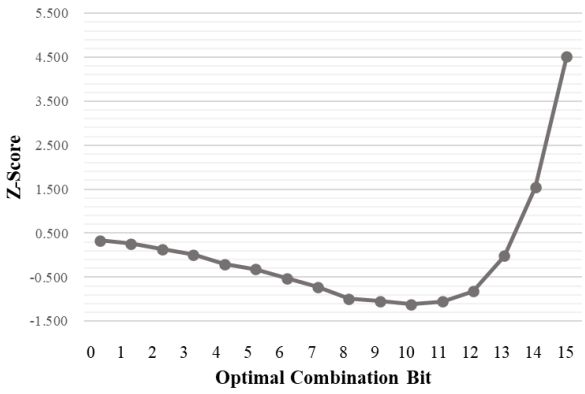

Figure 3. Z-Score by Optimal Combination Bit.

analysis is conducted, considering Mean Absolute Error (MAE), Mean Relative Error Distance (MRED), and circuit area as the three metrics. The circuit area is calculated as the total area for each circuit in Table 1, synthesized using the CMOS 28nm cell library and

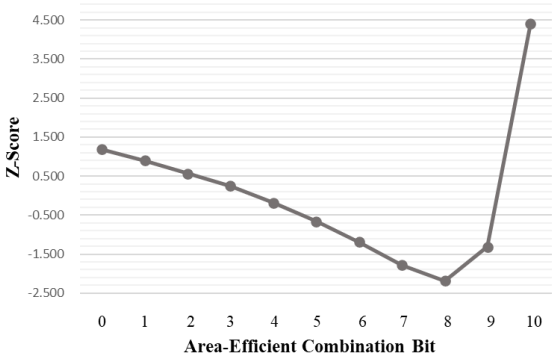

Figure 4. Z-Score by Area-Efficient Combination Bit.

a 1.1V operating voltage with Synopsys Design Compiler. MAE, MRED, and Z-Score are defined as in (6), (7), and (8).

$$\boldsymbol{MAE} = \frac{\mathbf{1}}{\boldsymbol{n}}\sum_{\boldsymbol{i=1}}^{\boldsymbol{n}}|\boldsymbol{x_i} - \boldsymbol{y_i}|, \tag{6}$$

$$\boldsymbol{MRED} = \frac{\mathbf{1}}{\boldsymbol{n}}\sum_{\boldsymbol{i=1}}^{\boldsymbol{n}}\frac{|\boldsymbol{x_i} - \boldsymbol{y_i}|}{\boldsymbol{x_i}}, \tag{7}$$

$$\boldsymbol{Z\text{-}Score} = \frac{\boldsymbol{x} - \boldsymbol{\mu}}{\boldsymbol{\sigma}}. \tag{8}$$

For a sample size of n, x represents the actual value, and y represents the approximate value. MAE and MRED are used as accuracy evaluation metrics. The Z-Score is the value obtained by transforming the original value x for each metric into a normal distribution. μ represents the mean, and σ represents the standard deviation. The overall metric is determined by adding the Z-Score for all three metrics, we define the least Z-Score is optimal. The experiments is conducted 10^6 times for each combination, and the metrics is averaged for use.

Table 3. AxPPA and EAxPPA Comparison

Matrix	AxPPA	EAxPPA
Circuit Area Saving	33.03%	68.35%
Power Saving	37.16%	64.02%
MAE	512.19	239.94
MRED	2.125%	0.986%

Table 2 presents the results of the experiments, based on the overall Z-Score, to determine the optimal approximate technique for design. The experimental results, combination [nor, cte-0, cte-0, cte-0, nor] exhibited the best performance.

C. Optimal Combination by Bit

The optimal number of bits for applying the most efficient approximate technique is determined through quantitative experiments. In the lower bits, the optimal combination approximate technique derived from B section is applied, and performance is compared based on the applied bits to derive the optimal ratio. The experimental results for design is as shown in Figure 3. The design is optimal performance when the approximate bits are set at 10 bits. Subsequently, based on the derived ratios, the part where the approximate technique is applied is further divided into three parts, the most area-efficient combination for the lower bits. The experimental results for design is shown in Figure 4. The results show optimal performance when the applied bits are set at 8 bits. Figure 2 represents EAxPPA structure verified through experiments proposed in this paper. Different approximate techniques are applied for each stage and bit, and these choices are the result of quantitative analysis through experiments. EAxPPA is designed based on the 16 bits Sklansky PPA. The upper 6 bits are the same as PPA, approximate technique is applied to the lower 10 bits. Upper 2 bits of these are applied the optimal approximate technique combination, lower 8 bits are applied area-efficient approximate technique combination.

4. Experimental Results based on ASIC

We conduct ASIC-based performance evaluations of the proposed structure. Using a CMOS 28nm standard cell library and a 1.1V operating voltage, we perform synthesis using Synopsys Design Compiler. We compare the results in terms of circuit area saving, power saving, MAE and MRED. For the comparison design, we implement PPA, AxPPA, and EAxPPA using Verilog. The approximate technique is applied to configure the bits to match the optimal configuration for the proposed structure. Table 3 represents the experimental results. The experimental results show that AxPPA achieved a 33.03% circuit area saving and a 37.16% power saving compared to PPA. On the other hand, EAxPPA achieved a remarkable 68.35% circuit area saving and a 64.02% power saving, demonstrating superior hardware performance. Comparing AxPPA and EAxPPA, EAxPPA exhibit a 52.74% circuit area saving effect and a 42.74% power saving effect compared to AxPPA. Additionally, it achieve a 53.15% reduction in MAE and a 53.6% reduction in MRED.

5. Conclusion

In this paper, we propose the EAxPPA, as a solution to reduce area and power consumption in PPA. We conduct experiments for all possible combinations of approximate techniques, using MAE, MRED, and area Z-Score as metrics, to derive the most efficient approximate technique combination and optimal approximate bit through quantitative analysis. we evaluate EAxPPA using ASIC synthesis, achieving a 72.25% circuit area saving and 69.34% power saving compare to PPA. Furthermore, compare to AxPPA, EAxPPA achieve a 52.74% circuit area saving, a 42.74% power saving, a 53.15% reduction in MAE , and a 53.6% reduction in MRED. Our method will be helpful in applying approximate techniques to other PPAs.

Acknowledgement

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (2022-0-01170), This research was supported by Basic Science Reserarch Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2021R1I1A3055806), and the EDA tool was supported by the IC Design Education Center(IDEC), Korea.

References

[1] D. Esposito, D. De Caro and A. G. M. Strollo, "Variable Latency Speculative Parallel Prefix Adders for Unsigned and Signed Operands," in IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 63, no. 8, pp. 1200-1209, Aug. 2016.

[2] M. M. A. d. Rosa, G. Paim, P. Ü. L. d. Costa, E. A. C. d. Costa, R. I. Soares and S. Bampi, "AxPPA: Approximate Parallel Prefix Adders," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 31, no. 1, pp. 17-28, Jan. 2023.

[3] P. M. Kogge and H. S. Stone, “A parallel algorithm for the efficient solution of a general class of recurrence equations,” IEEE Trans. Comput., vol. C-22, no. 8, pp. 786–793, Aug. 1973.

[4] R. P. Brent and H. T. Kung, “A regular layout for parallel adders,” IEEE Trans. Comput., vol. C-31, no. 3, pp. 260–264, Mar. 1982.

[5] J. Sklansky, “Conditional-sum addition logic,” IEEE Trans. Electron. Comput., vol. EC-9, no. 2, pp. 226–231, Jun. 1960.

[6] R. Ladner and M. Fischer, “Parallel prefix computation,” J. ACM, vol. 27, pp. 831–838, Oct. 1980.